

Undecidability of a Very Simple Modal Logic with Binding

Guillaume Hoffmann

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Universidad Blas Pascal

Universidad Nacional de Córdoba

`guillaume.hoffmann@conicet.gov.ar`

2015-08-14

Abstract

We show undecidability of the satisfiability problem of what is arguably the simplest non-sub-Boolean modal logic with an implicit notion of binding. This work enriches the series of existing results of undecidability of modal logics with binders, which started with Hybrid Logics and continued with Memory Logics.

1 Modal Logics, Names and Binders

Modal Logics are languages that are able to describe graphs from an internal perspective. When they are enriched with *nominals*, that is, propositional symbols that are true at only one state of the graph, they are called *Hybrid Logics* [4]. A nominal can unequivocally name a particular state. It was soon discovered that the ability to dynamically name states could lead to the ability to describe more interesting graphs. Goranko ([6]) introduced the *down-arrow binder*, with the syntax $\downarrow x.\varphi$ meaning “after naming the current state x , φ holds”. For instance, $\Box\downarrow x.\Diamond x$ means “all accessible states are reflexive”.

Hybrid Logics with binders are robustly undecidable, from the full $\mathcal{H}(@, \downarrow)$ language [5], to $\mathcal{H}(\downarrow)$ with only one relation and no propositional symbol beyond nominals [2]. Also, $\mathcal{H}(\downarrow)$ with propositional symbols, a single nominal and a single relation has the same fate [8].

Attempts have been made to find interesting decidable fragments of hybrid logics with binder. Some successful approaches involve restricting the class of models on which the satisfiability problem is specified: models of finite width [10], or models with a single relation and certain frame properties (S5, transitive, complete) [9].

On the other hand, a few syntactical restrictions have been successful. $\mathcal{H}(@, \downarrow) \setminus \Box\downarrow\Box$, the language obtained by removing formulas that contain a nesting of \Box , \downarrow and \Box , is decidable [10]. Also, only allowing one nominal, and restricting the depth between the binder and the nominal to 2 yields decidability [7].

2 Memory Logics

Memory logics [1, 3] are a distinct take on the binder. They are modal logics with the ability to *store* the current state of evaluation into some memory, and to consult whether the current state belongs to it. The two operators associated to these actions are $\textcircled{\mathsf{R}}$ and $\textcircled{\mathsf{K}}$. $\textcircled{\mathsf{R}}\varphi$ means “remember the current state and evaluate φ ” and $\textcircled{\mathsf{K}}$ means “this state is known”.

Given a model $\mathcal{M} = \langle W, R, V \rangle$, a state s and a memory S (which is a subset of W), the semantics of these operators is given by:

$$\begin{aligned} \mathcal{M}, S, s \models \textcircled{\mathsf{R}}\varphi & \quad \text{iff} \quad \mathcal{M}, S \cup \{s\}, s \models \varphi \\ \mathcal{M}, S, s \models \textcircled{\mathsf{K}} & \quad \text{iff} \quad s \in S. \end{aligned}$$

Let us take two pointed models that are bisimilar for the basic modal logic:



Assuming an empty initial memory, the formula $\textcircled{\mathsf{R}}\Diamond\textcircled{\mathsf{K}}$ is false on the left and true on the right. Whereas with hybrid logics we would have used an explicit name (e.g., with the formula $\downarrow a.\Diamond a$), memory logics do not involve any explicit binding process.

The satisfiability problem for the basic modal logic extended with $\textcircled{\mathsf{R}}$ and $\textcircled{\mathsf{K}}$ with initial empty memory is undecidable [3]. Another proposal is a memory logic without \Diamond and $\textcircled{\mathsf{R}}$, but with a remember-and-move operator $\langle r \rangle$ equivalent to $\textcircled{\mathsf{R}}\Diamond\varphi$. In this language, the operator $\textcircled{\mathsf{K}}$ identifies whether the current world has already been explored during the evaluation of the formula. It can be seen as an “I’ve already been here” operator. Again, satisfiability with an empty initial memory is undecidable in this simplified language.

Undecidability proofs for the aforementioned logics regularly involve encoding the tiling problem into the language of satisfiable formula [11]. Since sets of tiles are represented by propositional symbols, we propose to study a language that lacks them.

3 The $\mathcal{B}(\langle r \rangle, \textcircled{\mathsf{K}})$ fragment

We consider the language $\mathcal{B}(\langle r \rangle, \textcircled{\mathsf{K}})$. \mathcal{B} refers to the Boolean connectors, and $\langle r \rangle$ and $\textcircled{\mathsf{K}}$ are the only non-Boolean logical symbols. For readability sake, let us write \Diamond instead of $\langle r \rangle$. The syntax of $\mathcal{B}(\langle r \rangle, \textcircled{\mathsf{K}})$ is:

$$F := \neg F \mid F_1 \wedge F_2 \mid \Diamond F \mid \textcircled{\mathsf{K}}$$

We interpret formulas on Kripke frames $\mathcal{M} = \langle W, R \rangle$ made of a set of states W and a binary relation R on W , with some current evaluation state s and a set $S \subseteq W$ of visited states:

$$\begin{aligned} \mathcal{M}, S, s & \models \neg\varphi & \text{iff} & \mathcal{M}, S, s \not\models \varphi \\ \mathcal{M}, S, s & \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, S, s \models \varphi \text{ and } \mathcal{M}, S, s \models \psi \\ \mathcal{M}, S, s & \models \Diamond\varphi & \text{iff} & \text{there exists } t \in W \text{ such that} \\ & & & Rst \text{ and } \mathcal{M}, S \cup \{s\}, t \models \varphi \\ \mathcal{M}, S, s & \models \textcircled{\mathsf{K}} & \text{iff} & s \in S \end{aligned}$$

We define the notations \vee , \rightarrow and \Box as usual, and $\perp \equiv \mathbb{K} \wedge \neg \mathbb{K}$ and $\top \equiv \mathbb{K} \vee \neg \mathbb{K}$.

We will use the following notion of satisfiability:

Definition 1. A formula φ of $\mathcal{B}(\langle r \rangle, \mathbb{K})$ is satisfiable, or is in $\mathcal{B}(\langle r \rangle, \mathbb{K})$ -SAT, if there exists a model $\mathcal{M} = \langle W, R \rangle$ and $s \in W$ such that $\mathcal{M}, \emptyset, s \models \varphi$.

As a warm-up, let us see that we can define a formula satisfiable only in infinite models. Consider the formula *Inf* made of the following conjuncts:

1. s
2. $\Box \neg s$
3. $\Box \Box (\mathbb{K} \rightarrow s)$
4. $\Diamond \Diamond \mathbb{K}$
5. $\Box (\Diamond \top \rightarrow \Diamond (\neg \mathbb{K} \wedge \neg s))$
6. $\Box \Box (\neg s \rightarrow \Diamond (\mathbb{K} \wedge s \wedge \Diamond (\mathbb{K} \wedge \Box (\mathbb{K} \rightarrow s))))$
7. $\Box \Box (\neg s \rightarrow \Box (\neg s \rightarrow \Diamond (\mathbb{K} \wedge s \wedge \Box (\text{a-or-b} \rightarrow \Diamond c))))$

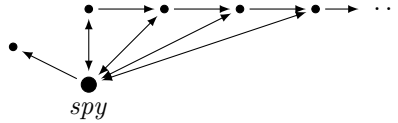
Which use the macros:

$$\begin{aligned} s &\equiv \Diamond \Box \perp \\ \text{a-or-b} &\equiv \mathbb{K} \wedge \Diamond (\mathbb{K} \wedge \neg s) \\ c &\equiv \mathbb{K} \wedge \Box (\mathbb{K} \rightarrow s) \end{aligned}$$

We call “spy” the evaluation state and review the conjuncts one by one, each time assuming all previous conjuncts also hold:

1. spy sees a dead-end
2. spy’s successors see no dead-ends, also implying spy is irreflexive
3. successors of spy are irreflexive
4. spy has a successor that has spy as a successor
5. if a successor of spy is not a dead-end, then it has a successor that is different from itself
6. all states of the chain are linked to spy in both directions, and the relation between chain states is asymmetric
7. relation between chain states is transitive (a-or-b and c are macros that help referring to successive states of the infinite chain)

As we require that the relation on chain states be serial (5), asymmetric (6), irreflexive (3) and transitive (7), we get an infinite chain of states (transitive links are not represented):



Theorem 1. *If there is a frame $\mathcal{M} = \langle W, R \rangle$ and $s \in W$ such that $\mathcal{M}, \emptyset, s \models \text{Inf}$ then W is infinite.*

Hence $\mathcal{B}(\langle r \rangle, \mathbb{K})$ lacks the finite model property. This does not necessarily imply that this logic is undecidable, but we are going to see that it is actually the case.

4 Undecidability of $\mathcal{B}(\langle r \rangle, \mathbb{K})$ -SAT

We adapt the proof of [8], where the tiling problem [11] is encoded into a description logic with the down-arrow binder, one nominal, and one relation.

We build a formula that is satisfiable only in models with the following components (see Figure 1):

- the *spy state* is where the formula is evaluated.
- *grid states* are direct successors of the spy state. A grid state is connected to other grid states, but also to switch and propositional states.
- *switch states* are in one-to-one correspondence with grid states, and help building the structure by simulating an explicit binding operator. A switch state sees the grid state that sees it.
- *propositional states* encode propositional symbols at every grid state, using chains of determined length. A grid state can see several propositional states, but a propositional state does not see its origin grid state.
- The spy, switch and propositional states see at least one dead-end state, while grid states do not.

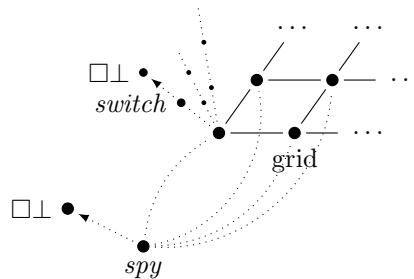


Figure 1: Model of a formula encoding some tiling problem, with details for the first grid state

Let us enumerate the conjuncts of the formula that encodes a given instance of the tiling problem.

The first formulas specify that the spy state sees a dead-end, is irreflexive, that all its successors are also irreflexive, that all grid states see the spy state, and that the relation between grid states is symmetric:

1. s

2. $\Box \neg s$
3. $\Box \Box (\mathbb{K} \rightarrow s)$
4. $\Box (\Diamond \top \rightarrow \Diamond \mathbb{K})$
5. $\Box \Box (\neg s \rightarrow \Diamond (\mathbb{K} \wedge \neg s))$

The following two formulas implement switch states for grid states accessible in one and two steps from spy:

6. $\Box (\Diamond \top \rightarrow \Diamond sw)$
7. $\Box \Box (\neg s \rightarrow \Diamond sw)$

With the macro:

$$\begin{aligned} sw &\equiv \neg \mathbb{K} \wedge s \\ &\wedge \Diamond (\mathbb{K} \wedge \neg s) \\ &\wedge \Box (\Diamond \top \rightarrow (\mathbb{K} \wedge \neg s \wedge \Box ((s \wedge \Diamond \mathbb{K}) \rightarrow \mathbb{K}))) \end{aligned}$$

Each grid state sees to its own switch state, which sees a dead-end and sees the grid state that sees it. Switch states help us simulate the \mathbb{R} operator: to “remember” a particular grid state, we visit its associated switch state and come back. Note that formula (7) depends on formula (6) to implement switches on grid states distant from two steps from spy.

We use the following macro to “remember” the current grid state:

$$\mathbb{R}' \varphi \equiv \Box (\neg \mathbb{K} \wedge s \rightarrow \Box (\mathbb{K} \rightarrow \varphi))$$

And to check that a grid state has been remembered:

$$\mathbb{K}' \equiv \mathbb{K} \wedge \Box ((s \wedge \Diamond \mathbb{K}) \rightarrow \mathbb{K})$$

Now the following formula makes the spy state see every grid state:

8. $\Box \Box (\neg s \rightarrow \mathbb{R}' \Box \Box (\mathbb{K} \wedge s \rightarrow \Diamond \mathbb{K}'))$

We encode the tiling problem with only one relation, by combining a symmetric relation with “propositional macros” to represent two “meta-relations” *up* and *right*, as shown in Figure 2. In what follows, we use the notations i , $s(i)$ and $p(i)$, with:

$$\begin{aligned} i &\in \{0, 1, 2\} \\ s(i) &= (i + 1) \bmod 3 \\ p(i) &= (i + 2) \bmod 3 \end{aligned}$$

Every grid state has an “up”- and a “right”-successor:

9. $\bigwedge_{i \in \{0, 1, 2\}} \Box (\Diamond \top \wedge i \rightarrow \Diamond (\neg s \wedge u \wedge \Diamond (\neg s \wedge s(i))))$
10. $\bigwedge_{i \in \{0, 1, 2\}} \Box (\Diamond \top \wedge i \rightarrow \Diamond (\neg s \wedge r \wedge \Diamond (\neg s \wedge p(i))))$

“Up” and “right” are irreflexive and functional:

11. $\bigwedge_{i \in \{0, 1, 2\}} \Box (\Diamond \top \wedge s(i) \rightarrow \mathbb{R}' \Box (\neg s \wedge u \rightarrow \Box (\neg s \wedge i \rightarrow \neg \mathbb{K}) \wedge \Box (\neg s \wedge u \rightarrow \Box (\neg s \wedge s(i) \rightarrow \mathbb{K}')))))$

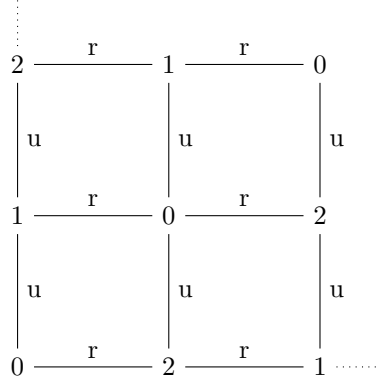


Figure 2: Organization of the grid

12. $\bigwedge_{i \in \{0,1,2\}} \Box(\Diamond \top \wedge p(i) \rightarrow \textcircled{\mathsf{R}}' \Box(\neg s \wedge r \rightarrow \Box(\neg s \wedge i \rightarrow \neg \textcircled{\mathsf{K}} \wedge \Box(\neg s \wedge r \rightarrow \Box(\neg s \wedge p(i) \rightarrow \textcircled{\mathsf{K}}')))))$

We finish with the confluence property as shown in Figure 3:

13. $\bigwedge_{i \in \{0,1,2\}} \Box(\Diamond \top \wedge s(i) \rightarrow \textcircled{\mathsf{R}}' \Box(\neg s \wedge u \rightarrow \Box(\neg s \wedge i \rightarrow \Box(\neg s \wedge r \rightarrow \Box(\neg s \wedge p(i) \rightarrow \Diamond(\neg s \wedge u \wedge \Diamond(\neg s \wedge i \wedge \Diamond(\neg s \wedge r \wedge \Diamond(\neg s \wedge s(i) \wedge \textcircled{\mathsf{K}}'))))))))$

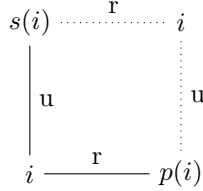


Figure 3: Confluence property

Now, we encode the actual tiling problem. Given $T = T_1, \dots, T_k$ a set of tile types, we represent each type T_i as a macro t_i . At every point of the grid, one and only one tile holds:

14. $\bigwedge_{i \in \{0,1,2\}} \Box(\Diamond \top \wedge i \rightarrow (\bigvee_{1 \leq n \leq k} t_n) \wedge (\bigwedge_{1 \leq n < m \leq k} \neg(t_n \wedge t_m)))$

Then, tile matching:

15. $\bigwedge_{i \in \{0,1,2\}, 1 \leq n \leq k} \Box(\Diamond \top \wedge i \wedge t_n \rightarrow \Box(\neg s \wedge u \rightarrow \Box(\neg s \wedge s(i) \rightarrow \bigvee \{t_m \mid \text{top}(T_n) = \text{bottom}(T_m)\}))))$
16. $\bigwedge_{i \in \{0,1,2\}, 1 \leq n \leq k} \Box(\Diamond \top \wedge i \wedge t_n \rightarrow \Box(\neg s \wedge r \rightarrow \Box(\neg s \wedge p(i) \rightarrow \bigvee \{t_m \mid \text{right}(T_n) = \text{left}(T_m)\}))))$

We initialize the grid with a first state:

17. $\Diamond(\Diamond \top \wedge 0)$

Finally, we assign a distinct natural number p to every propositional macro that appear in formulas 9–17, and rewrite it to:

$$p \equiv \Diamond(s \wedge \neg \mathbb{K} \wedge \Box \neg \mathbb{K} \wedge \overbrace{\Diamond \Diamond \dots \Diamond}^{p \text{ times}} \Box \perp)$$

We call “propositional states” the states accessible from grid states where $s \wedge \neg \mathbb{K} \wedge \Box \neg \mathbb{K}$ holds. Hence, propositional states do not see the grid point that see them. Each “propositional symbol” that should be true at a grid state is represented by a propositional state that sees a dead-end, does not see the grid point where it came from, and sees the beginning of a chain of determined length. Propositional states and their corresponding chains are not seen by the spy state (see formula (8)). One can check that negating such a macro does not interfere with the structure of the grid:

$$\neg p \equiv \Box(s \wedge \neg \mathbb{K} \wedge \Box \neg \mathbb{K} \rightarrow \overbrace{\Diamond \Diamond \dots \Diamond}^{p \text{ times}} \Box \perp)$$

Lemma 1. *Given a tiling problem $T = T_1 \dots T_k$, let $\text{Grid}(T)$ be the conjunction of formulas 1–17 rewritten with the propositional encoding previously shown.*

T tiles the grid if, and only if, $\text{Grid}(T) \in \mathcal{B}(\langle r \rangle, \mathbb{K})\text{-SAT}$.

Theorem 2. $\mathcal{B}(\langle r \rangle, \mathbb{K})\text{-SAT}$ is undecidable.

5 Closing Remarks

We showed that $\mathcal{B}(\langle r \rangle, \mathbb{K})$, arguably the simplest non-sub-Boolean modal logic with binding, has an undecidable satisfiability problem. This result can be reused to show undecidability of the satisfiability problem of most memory logics and hybrid logics with binders, by means of simple satisfiability-preserving translations. For instance the following translation from $\mathcal{B}(\langle r \rangle, \mathbb{K})\text{-SAT}$ to $\mathcal{H}(\downarrow)\text{-SAT}$ [3]:

$$\begin{aligned} \text{Tr}_N(\neg \varphi) &= \neg \text{Tr}_N(\varphi) \\ \text{Tr}_N(\varphi \wedge \psi) &= \text{Tr}_N(\varphi) \wedge \text{Tr}_N(\psi) \\ \text{Tr}_N(\Diamond \varphi) &= \downarrow i. \text{Tr}_{N \cup \{i\}}(\varphi) \text{ with } i \notin N \\ \text{Tr}_N(\mathbb{K}) &= \bigvee_{i \in N} i \end{aligned}$$

where N is a subset of the set of nominals, initially empty.

Taking as a starting point the basic hybrid logic with binder $\mathcal{H}(@, \downarrow)$, the undecidability results for modal logics with binding have progressed in two directions. First, by maintaining the ability to bind an arbitrary number of states, while decreasing the expressive power of the language. This is how we went from $\mathcal{H}(@, \downarrow)$, to $\mathcal{H}(\downarrow)$ without non-nominal propositional symbols [2], to memory logics with \mathbb{T} and \mathbb{K} [3], to memory logics with $\langle r \rangle$ instead of \mathbb{T} and \Diamond , and finally, to the memory logic presented in this article.

The second direction has been to restrict the number of bindable states to only one [8]. This approach actually removes the differences between hybrid and memory logics (the latter of which were not yet invented at the time of the result).

It is not clear how to transpose this last approach to the logic presented here. One way would be to restrict the memory to the last n visited states. But this involves giving up on the simplicity of the very definition of the problem. Hence it seems probable that the quest for pushing the undecidable frontier will continue within a different setting.

References

- [1] C. Areces. Hybrid logics: The old and the new. In X. Arrazola and J. Larrazabal, editors, *Proceedings of LogKCA-07*, pages 15–29, San Sebastian, Spain, 2007.
- [2] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321, Madrid, Spain, 1999. Springer. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999.
- [3] C. Areces, D. Figueira, S. Figueira, and S. Mera. The expressive power of memory logics. *Review of Symbolic Logic*, 4(2):290–318, 2011.
- [4] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 821–868. Elsevier, 2006.
- [5] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.
- [6] V. Goranko. Temporal logic with reference pointers. In *Temporal logic*, volume 827 of *LNCS*, pages 133–148. Berlin: Springer, 1994.
- [7] Daniel Gorín and Lutz Schröder. Narcissists are easy, stepmothers are hard. In Lars Birkedal, editor, *Foundations of Software Science and Computational Structures*, volume 7213 of *Lecture Notes in Computer Science*, pages 240–254. Springer Berlin Heidelberg, 2012.
- [8] Maarten Marx. Narcissists, stepmothers and spies. In *In Proc. of DL02, volume 53. CEUR*, 2002.
- [9] T. Schneider. *The Complexity of Hybrid Logics over Restricted Frame Classes*. PhD thesis, University of Jena, 2007.
- [10] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In L. Ong, editor, *Proceedings of Computer Science Logic 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer Verlag, 2005.
- [11] Peter Van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. Marcel Dekker Inc, 1997.